Системы технического зрения

ГИСТОГРАММЫ. ЦВЕТОВОЕ ПРОСТРАНСТВО HSV

Opencv_first_steps

```cpp
#include <iostream>
#include <string>
#include <opencv2/opencv.hpp>

using namespace std;


int main()
{
    cv::Mat image = cv::imread("surface.jpg");      //  Считываем файл surface.jpg
    if (!image.data)                                //  Проверяем, удалось ли его  открыть
        return -1;                                  //  Если нет - оставшуюся часть выполнять не будем

    cv::imshow("Test", image);                      //  Выведем результат на экран
    cv::waitKey();                                  //  И подождём нажатия на кнопку
    std::vector<cv::Mat> planes;                    //  Теперь я сформирую вектор из отделных матриц
    cv::split (image, planes);                      //  И разделю исходный файл на отдельные каналы
                                                    //  Каналы соответствуют цвету - blue, green, red
    for (int i=0; i<planes.size(); i++)             //  А теперь выведем их на экран по отдельности
    {
        std::string name;
        switch (i)                                  //  Переключатель case - несколько удобнее, чем
        {                                           //  несколько if
            case 2 :
                name = "red";
                break;
            case 1 :
                name = "green";
                break;
            case 0 :
                name = "blue";
        }
        imshow (name, planes[i]);
    }
    cv::waitKey();
```

Opencv_first_steps

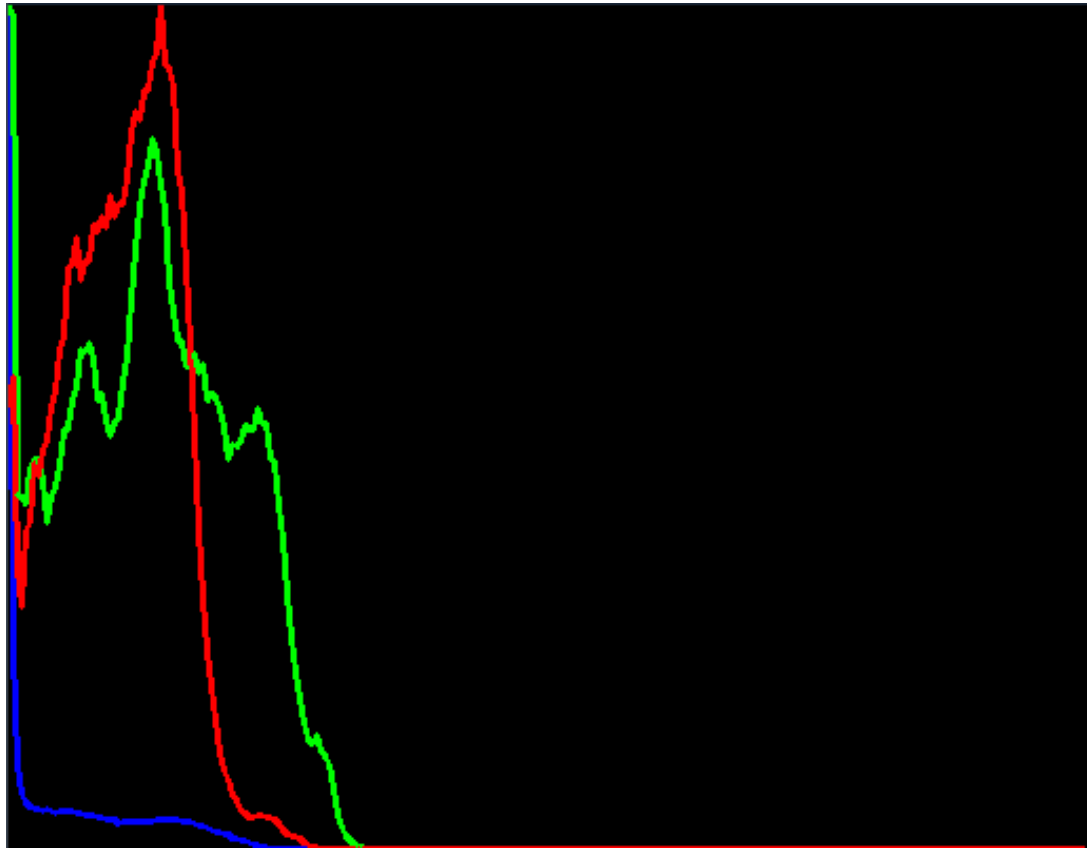
```
int histSize = 256;                                    //  У меня в каждом канале 8-битная кодировка цвета
float range[]= {0, 255};                               //  т.е. кодируется 2^8^3 = 16 777 216 цветов
const float* pRange = range;
int channels[] = {0};

cv::Mat rHist, bHist, gHist;
//calcHist (указатель на входные данные ,число каналов, массив номеров каналов, cv::Mat(),
//  результат, число каналов  результатата, размер гистограмы, диапазон);
cv::calcHist(&planes[0],1,(int*)channels,cv::Mat(), bHist, 1, &histSize, &(pRange));
cv::calcHist(&planes[1],1,(int*)channels,cv::Mat(), gHist, 1, &histSize, &(pRange));
cv::calcHist(&planes[2],1,(int*)channels,cv::Mat(), rHist, 1, &histSize, &(pRange));

int hist_w = 512, hist_h=400;
int bin_w = cvRound((double)hist_w / histSize);
cv::Mat histImage(hist_h, hist_w, CV_8UC3, cv::Scalar(0,0,0));
//  Теперь приведём все гистограмы к одному масштабу
normalize(bHist, bHist, 0, histImage.rows, cv::NORM_MINMAX, -1, cv::Mat());
normalize(gHist, gHist, 0, histImage.rows, cv::NORM_MINMAX, -1, cv::Mat());
normalize(rHist, rHist, 0, histImage.rows, cv::NORM_MINMAX, -1, cv::Mat());

for (int i=1; i<histSize; i++)
{
    //  Нарисуем все гисторграмы элементами линий
    cv::line (histImage, cv::Point(bin_w*(i-1), hist_h-cvRound(bHist.at<float>(i-1)) ),
              cv::Point(bin_w*(i), hist_h-cvRound(bHist.at<float>(i)) ),
              cv::Scalar(255,0,0), 2,8,0);
    cv::line (histImage, cv::Point(bin_w*(i-1), hist_h-cvRound(gHist.at<float>(i-1)) ),
              cv::Point(bin_w*(i), hist_h-cvRound(gHist.at<float>(i)) ),
              cv::Scalar(0,255,0), 2,8,0);
    cv::line (histImage, cv::Point(bin_w*(i-1), hist_h-cvRound(rHist.at<float>(i-1)) ),
              cv::Point(bin_w*(i), hist_h-cvRound(rHist.at<float>(i)) ),
              cv::Scalar(0,0,255), 2,8,0);
}

cv::imshow ("hist",histImage);
cv::waitKey();


return 0;
```

Opencv_first_steps: подключаемы библиотеки

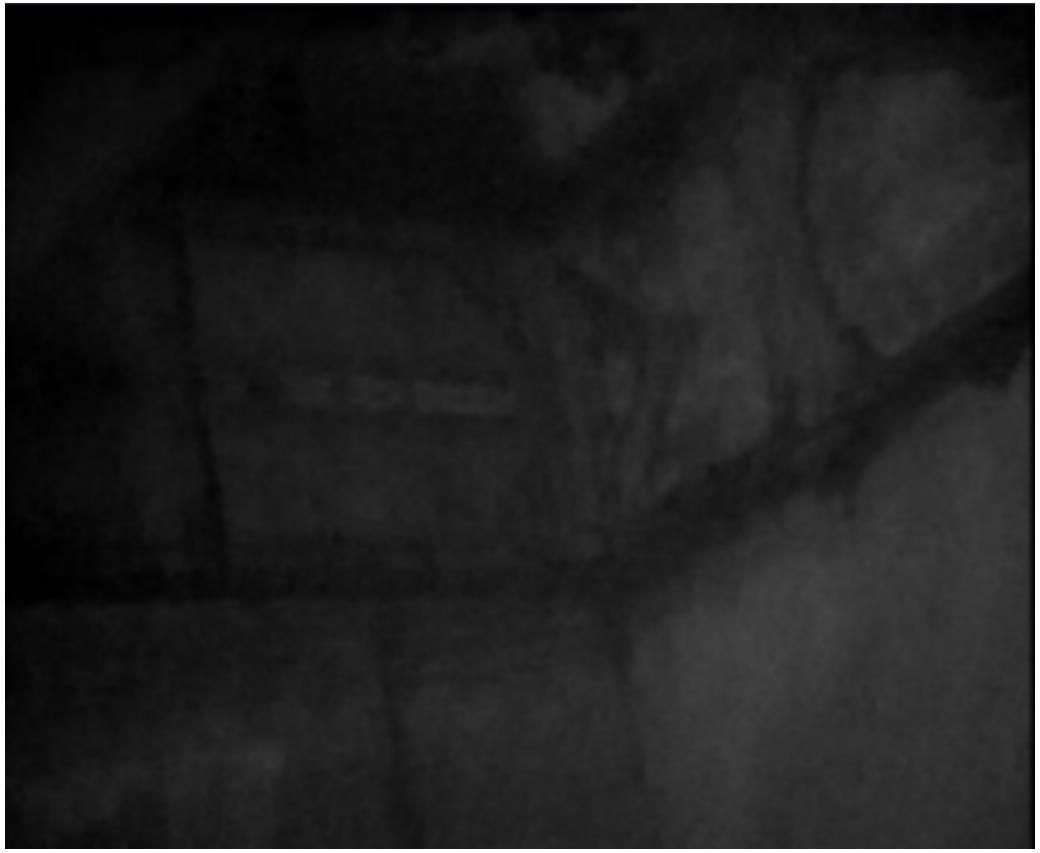
opencv first steps.pro

```
1  TEMPLATE = app
2  CONFIG += console
3  CONFIG -= app_bundle
4  CONFIG -= qt
5
6  SOURCES += main.cpp
7  #   А вот здесь я подключаю библиотеки OpenCV
8  LIBS += -L/usr/lib/i386-linux-gnu   \
9          -lopencv_core   \
10         -lopencv_highgui \
11         -lopencv_imgproc
```

Гистограммы: разложение по каналам



Попробуем улучшить изображение!



Opencv_histogram

```
#include <iostream>
#include <string>
#include <opencv2/opencv.hpp>

using namespace std;

cv::Mat makeHistogram (cv::Mat& image);
cv::Mat brightnessEnch (cv::Mat& greyScale);

int main()
{
    cv::Mat image = cv::imread("surface.jpg");
//    cv::Mat image = cv::imread("test_bad.jpg");
    if (!image.data)
        return -1;

    cv::imshow("Test", image);
    cv::Mat greyScale;
    cvtColor(image, greyScale, CV_BGR2GRAY);
    cv::GaussianBlur(greyScale,greyScale,cv::Size(5,5),3);
    imshow ("greyScale", greyScale);
    cv::waitKey();

    cv::Mat histImage = makeHistogram(greyScale);
    cv::imshow ("Histogram",histImage);
    cv::waitKey();

    cv::Mat newGreyScale = brightnessEnch(greyScale);
    histImage = makeHistogram(newGreyScale);
    cv::imshow ("New",newGreyScale);
    cv::imshow ("New Histogram",histImage);
    cv::waitKey();

    return 0;
}
```

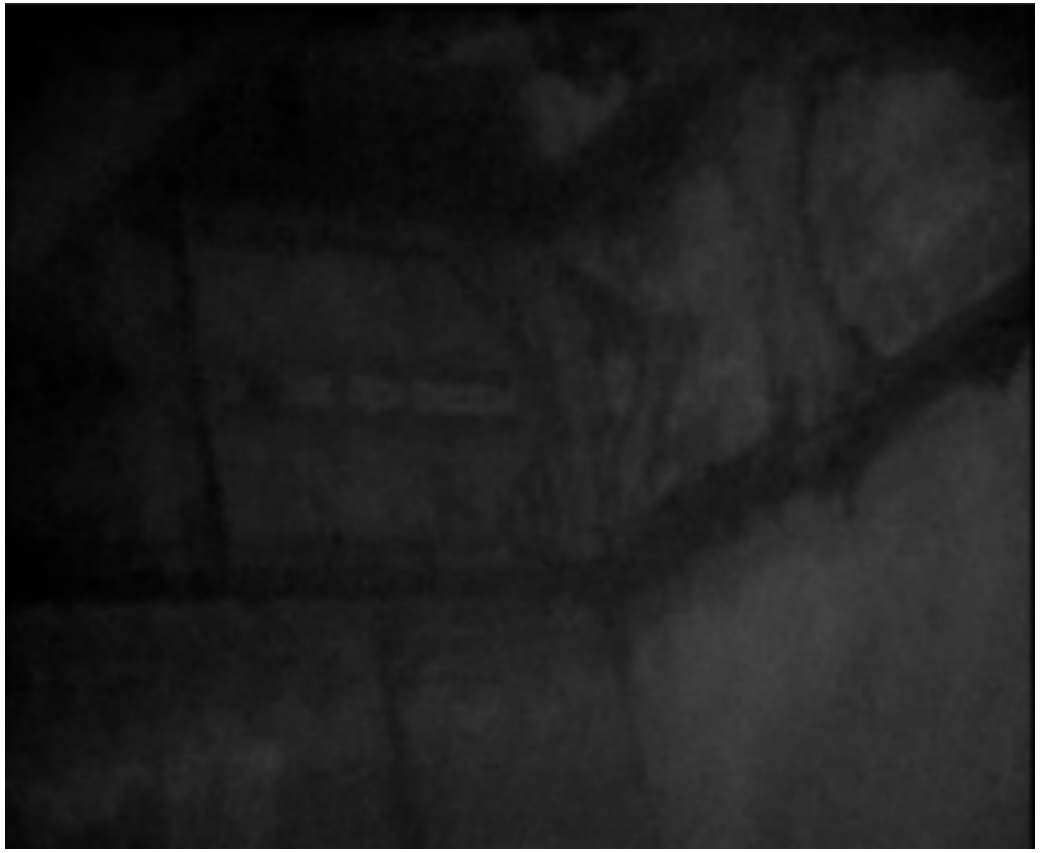
Opencv_histogram

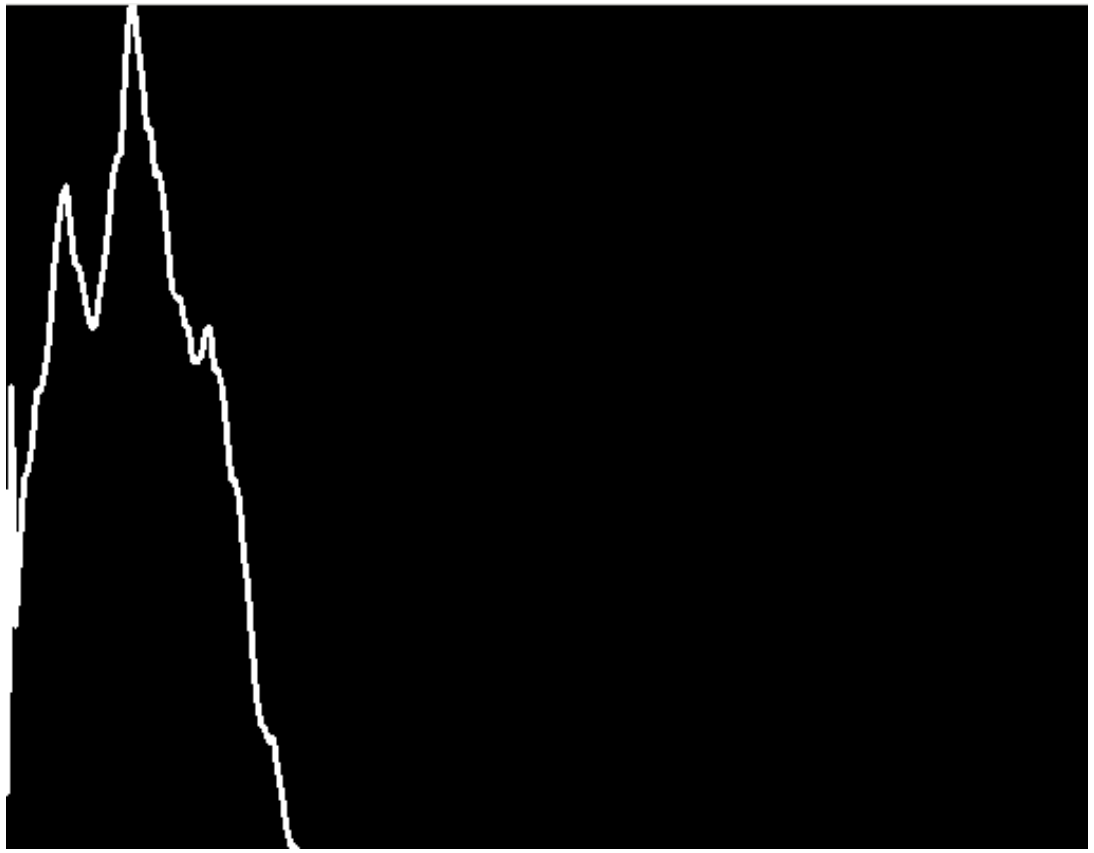
```
36
37  cv::Mat makeHistogram (cv::Mat& image)
38  {
39      int histSize = 256;
40      float range[]= {0, 255};
41      const float* pRange = range;
42      int channels[] = {0};
43      cv::Mat hist;
44
45      cv::calcHist(&image,1,(int*)channels,cv::Mat(), hist, 1, &histSize, &(pRange));
46      int hist_w = 512, hist_h=400;
47      int bin_w = cvRound((double)hist_w / histSize);
48      cv::Mat histImage(hist_h, hist_w, CV_8UC3, cv::Scalar(0,0,0));
49      normalize(hist, hist, 0, histImage.rows, cv::NORM_MINMAX, -1, cv::Mat());
50      for (int i=1; i<histSize; i++)
51      {
52          cv::line (histImage, cv::Point(bin_w*(i-1), hist_h-cvRound(hist.at<float>(i-1)) ),
53                    cv::Point(bin_w*(i), hist_h-cvRound(hist.at<float>(i)) ),
54                    cv::Scalar(255,255,255), 2,8,0);
55      }
56      return histImage;
57  }
```

Opencv_histogram

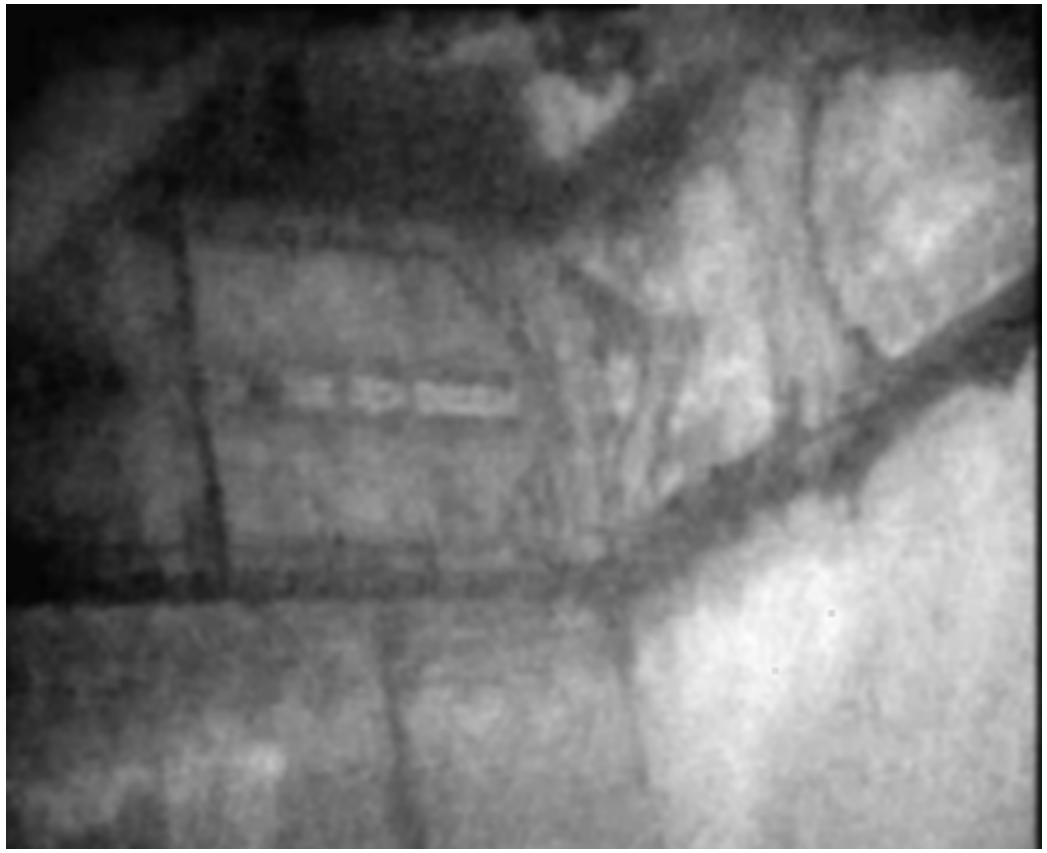
```
59  cv::Mat brightnessEnch (cv::Mat& greyScale)
60  {
61      double minValue, maxValue;
62      cv::minMaxLoc(greyScale, &minValue, &maxValue);
63      double deltaStep = 255./(maxValue - minValue), value=0;
64      cv::Mat lookUpTable (1, 256, CV_8U);
65      cv::Mat newGreyScale;
66      for (int i=0; i<255; i++)
67      {
68          if (i<minValue)
69              lookUpTable.at<unsigned char>(i) = minValue;
70          if ( i>=minValue && i<maxValue)
71          {
72              value += deltaStep;
73              lookUpTable.at<unsigned char>(i) = (unsigned char) value;
74          }
75          if (i>=maxValue)
76              lookUpTable.at<unsigned char>(i) = maxValue;
77      }
78      cv::LUT(greyScale, lookUpTable, newGreyScale);
79      cv::GaussianBlur(newGreyScale,newGreyScale,cv::Size(5,5),3);
80      return newGreyScale;
81  }
```

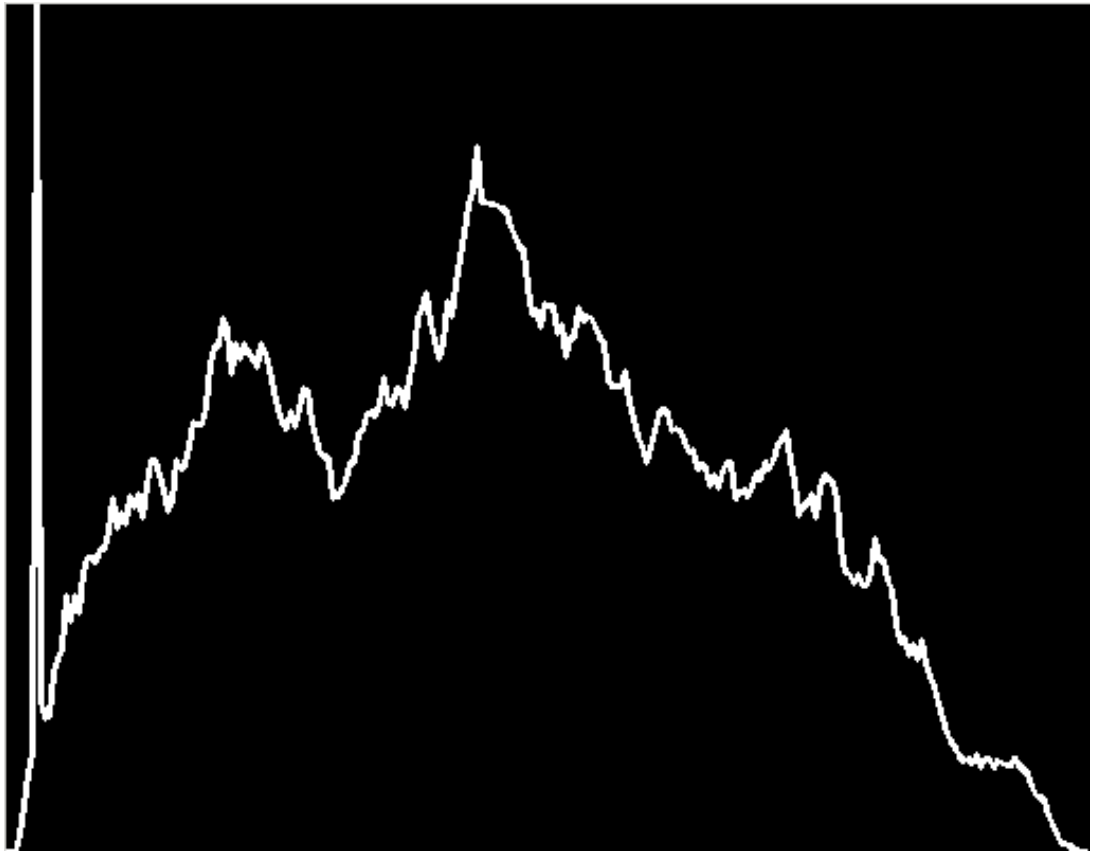
Гистограммы: полутоновое изображение



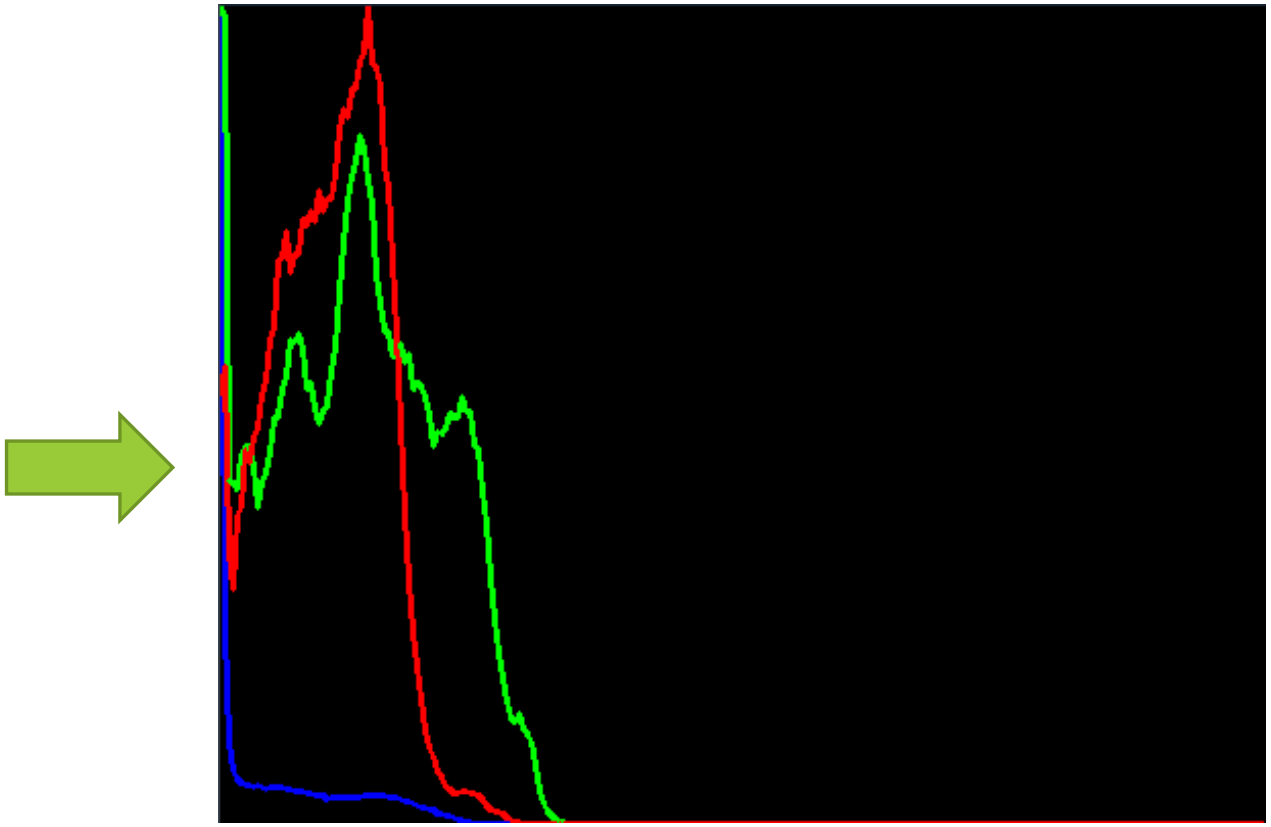


Гистограммы: полутоновое изображение

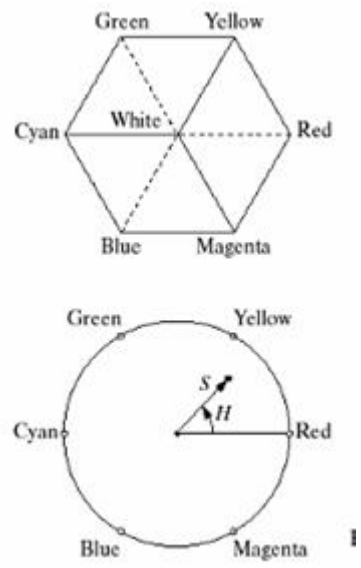


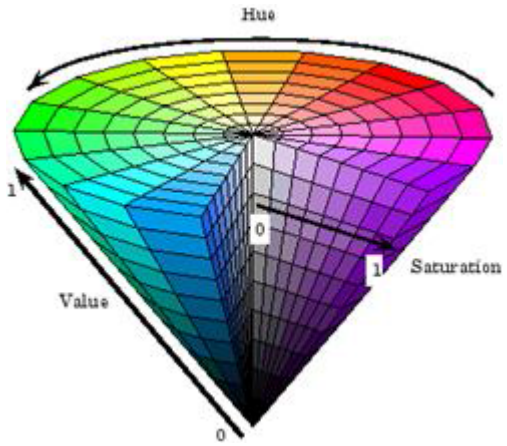


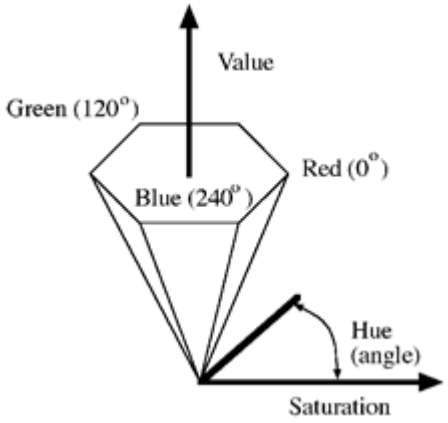
Гистограммы: а что же с цветом?



Гистограммы: HSV







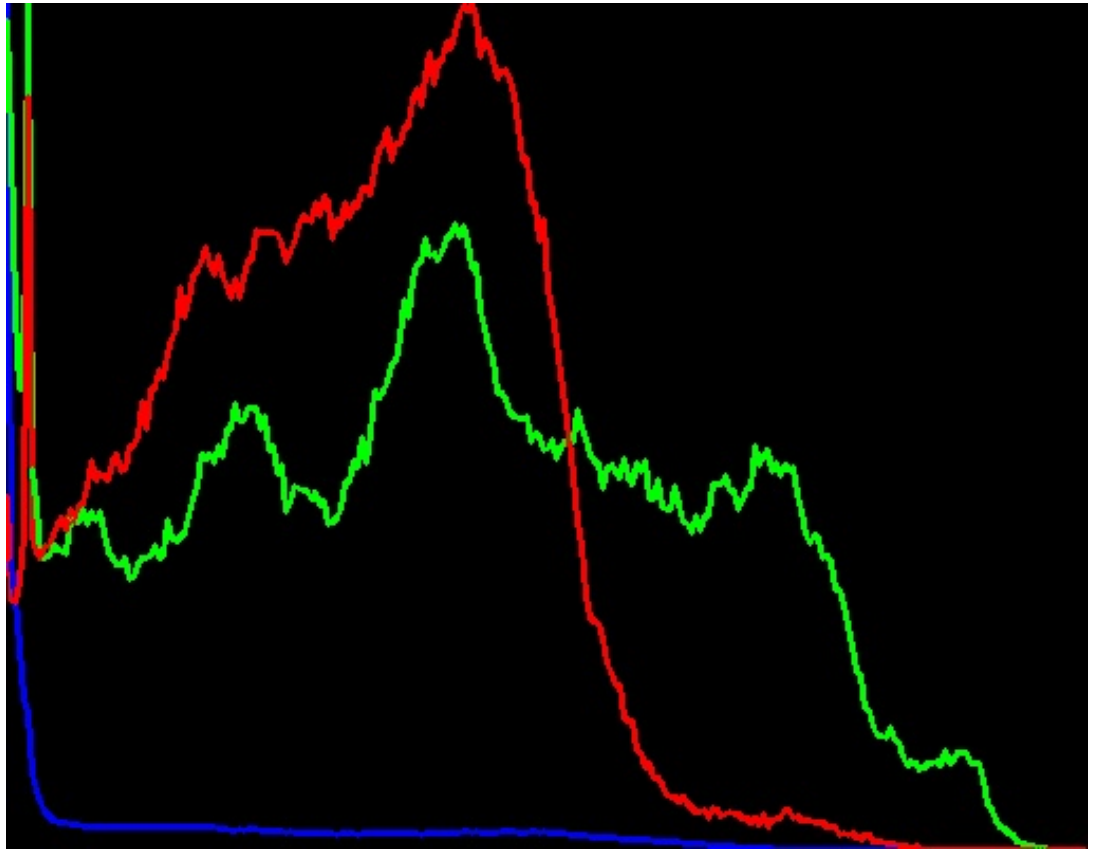
Hue (Тон), Saturation(Насыщенность),
Value (Интенсивность)
Очевидно, что в данном случае, нас интересует Value!

Opencv_hsv

```
#include <iostream>
#include <string>
#include <opencv2/opencv.hpp>

using namespace std;

cv::Mat makeHistogram (cv::Mat& image);
cv::Mat brightnessEnch (cv::Mat& greyScale);

int main()
{
    cv::Mat image = cv::imread("surface.jpg"), hsvImage;
    if (!image.data)
        return -1;

    cv::imshow("Test", image);
    std::vector<cv::Mat> planes;
    cout << "Total number of channels " << image.channels() << endl;
    cvtColor(image, hsvImage, CV_RGB2HSV);
    cv::split(hsvImage, planes);
    cv::Mat value = planes[2];

//    imshow ("Value", value);
    cv::waitKey();

    cv::Mat histImage = makeHistogram(value);
//    cv::imshow ("Histogram",histImage);
//    cv::waitKey();

    cv::Mat newValue = brightnessEnch(value);
//    cv::imshow ("New value",planes[2]);
    cv::Mat newImage =hsvImage;//(image.rows, image.cols, CV_8UC3);
    cv::Mat newPlanes [] = {planes[0], planes[1], newValue};
//    int fromTo [] = {0,0, 1,1, 2,2};
    cv::merge(newPlanes, 3, newImage);

    histImage = makeHistogram(newValue);
    cvtColor(hsvImage, hsvImage, CV_HSV2RGB);
    cv::imshow ("New",newImage);
//    cv::imshow ("New Histogram",histImage);
    cv::waitKey();

    return 0;
}
```

Гистограммы: а что же с цветом?

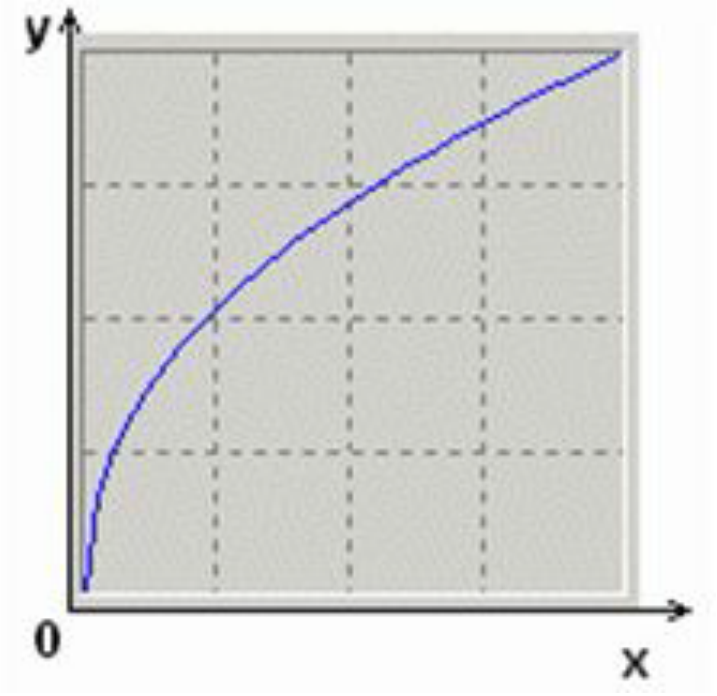
Гистограммы: а что же с цветом?



Гистограммы

Гистограммы

Нелинейная коррекция



Шумоподавление

Original | Salt and pepper noise

Impulse noise | Gaussian noise

Соль и перец: случайные черные и белые пиксели

• **Импульсный:** случайные белые пиксели

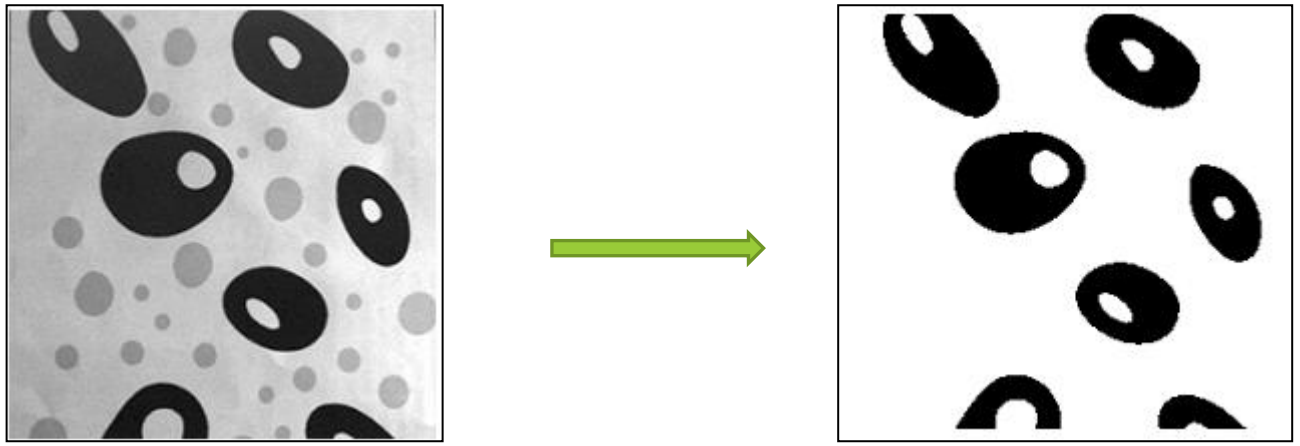
• **Гауссов: колебания** яркости, распределенные по нормальному закону

Бинарное изображение

Уменьшение объёма информации

Упрощение её обработки

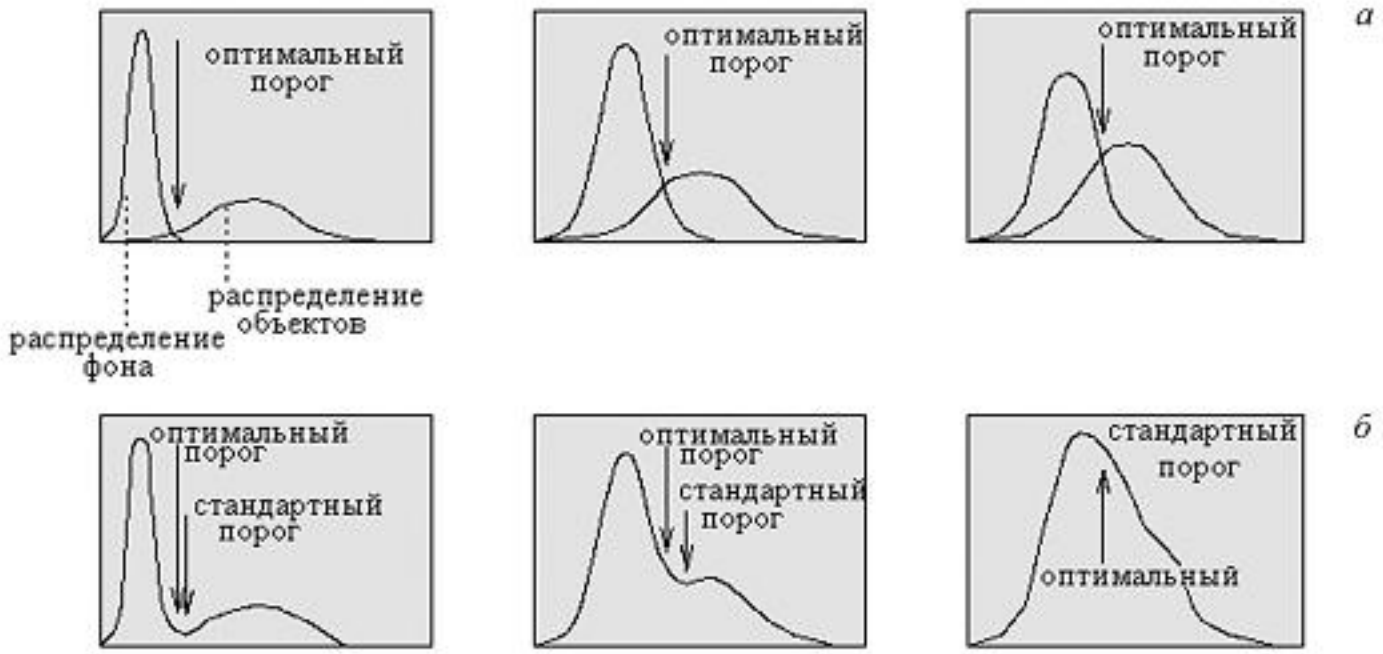
Как её произвести?



Бинарное изображение

На основе обработки гистограмы яркостей

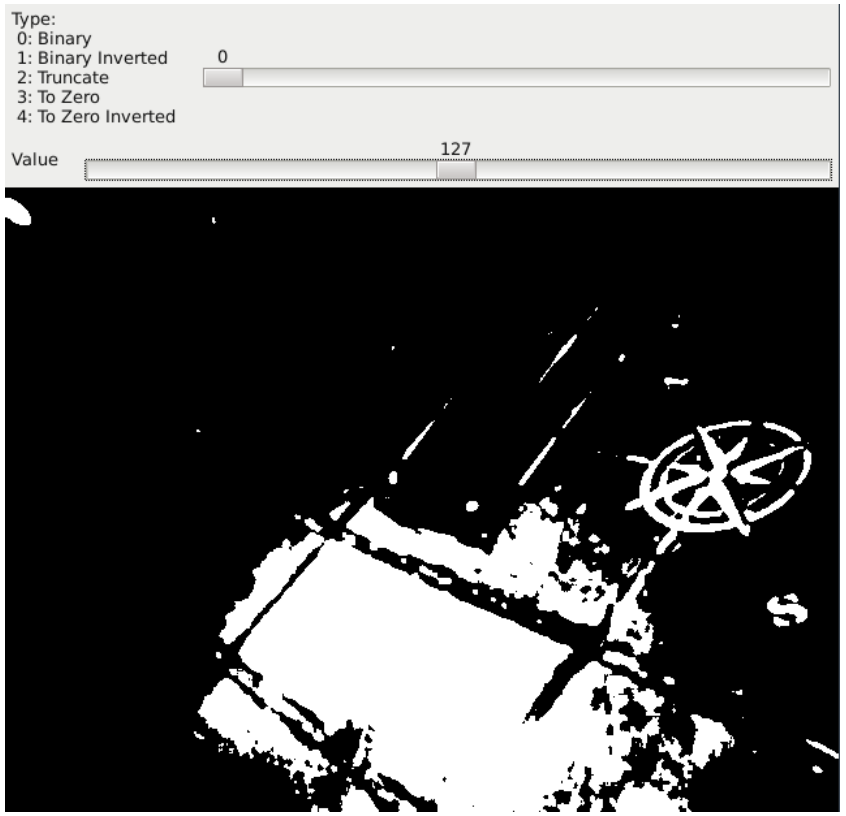
Выбор конкретного положения?



opencv_binarization

ОРИГИНАЛ

РЕЗУЛЬТАТ



Задание №4

Мы попробуем распознать код товара и место, где он лежит

Для этого на первом этапе нам нужно сделать изображение бинарным и ”почистить» от шумов